NRL Memorandum Report 2522

# Signal Processing Element Functional Description

## Part 2 (Preliminary) - Signal Processing Arithmetic Unit

WILLIAM R. SMITH AND HAROLD H. SMITH

*Information Systems Group*
*Office of the Associate Director of Research*
*for Electronics*



October 1972



NAVAL RESEARCH LABORATORY
Washington, D.C.

54

Unclassified

Security Classification

**DOCUMENT CONTROL DATA - R & D**

*(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)*

| 1 ORIGINATING ACTIVITY (Corporate author) | 2a. REPORT SECURITY CLASSIFICATION |
|---|---|
| Naval Research Laboratory<br>Washington, D.C. 20390 | Unclassified |
| | 2b. GROUP |

3 REPORT TITLE

**Signal Processing Element Functional Description**
**Part 2 (Preliminary) — Signal Processing Arithmetic Unit**

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

An interim report on continuing NRL problems.

5 AUTHOR(S) *(First name, middle initial, last name)*

William R. Smith and Harold H. Smith

| 6 REPORT DATE | 7a. TOTAL NO OF PAGES | 7b. NO. OF REFS |
|---|---|---|
| October 1972 | 52 | |

| 8a. CONTRACT OR GRANT NO. | 9a. ORIGINATOR'S REPORT NUMBER(S) |
|---|---|
| NRL Problems B02-06 and B02-10 | NRL Memorandum Report 2522 |
| b. PROJECT NO. 3F21-241-601 (formerly WF-15-241-601) | |
| c. 2F00-241-601, XF-53-241-003, and | 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* |
| d. XF-21-241-015-K152 | |

10 DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

| 11 SUPPLEMENTARY NOTES | 12. SPONSORING MILITARY ACTIVITY |
|---|---|
| | Department of the Navy (Naval Air Systems Command and Naval Command, Control, Communications and Electronics Command,) ~~Washington, D.C. 20360~~ |

13 ABSTRACT

The NRL Signal Processing Element (SPE) is being developed to provide a high-performance signal processing facility for radar, sonar, and communication systems. It is intended to be compatible with the Navy's All Applications Digital Computer (AADC). The SPE consists of four major subsystems; a Microprogrammed Control Unit (MCU), a Buffer Store and Storage Control Unit (SPU), a Signal Processing Arithmetic Unit (SPAU) and Input/Output (I/O) units.

The MCU serves as system supervisor and data organizer for the SPAU and other I/O devices. The MCU includes a 64-bit Control Store, two local stores, an arithmetic element, two busses to buffer memory, an unbuffered byte channel, and priority interrupt system. Its cycle time is 150 nanoseconds (nsec).

The buffer store and SCU consists of eight fixed-priority Direct Memory Access (DMA), or Buffered, Channels communicating on a 150-nsec cycle basis with up to eight 32-bit-by-4K memories.

The SPAU is a highly parallel, high-speed arithmetic unit capable of performing complex signal processing operations such as FFT and recursive filtering. It is microprogrammed for flexibility in adapting signal processing algorithms. Four multiplies and six additions can be performed in 300 nsec.

SPE I/O and internal communications are provided by DMA buffer data channels, an unbuffered byte channel, and a priority interrupt system. The unbuffered byte channel called the Z Bus communicates both data and control information to all I/O devices. The unbuffered channel burst data rate is 2 million 16-bit words per second.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

IA

Unclassified

Security Classification

| 14 KEY WORDS | LINK A | | LINK B | | LINK C | |
|---|---|---|---|---|---|---|
| | ROLE | WT | ROLE | WT | ROLE | WT |
| Signal Processing Element | | | | | | |
| Signal Processing Arithmetic Unit (SPAU) | | | | | | |
| All Applications Digital Computer (AADC) | | | | | | |
| Parallel Processing | | | | | | |
| Pipelining | | | | | | |
| Microprogrammed Control Unit (MCU) | | | | | | |



I k

CONTENTS

## ABSTRACT

The NRL Signal Processing Element (SPE) is being developed to provide a high-performance signal processing facility for radar, sonar, and communication systems. It is intended to be compatible with the Navy's All Applications Digital Computer (AADC). The SPE consists of four major subsystems; a Microprogrammed Control Unit (MCU), a Buffer Store and Storage Control Unit (SCU), a Signal Processing Arithmetic Unit (SPAU), and Input/Output (I/O) units.

The MCU serves as system supervisor and data organizer for the SPAU and other I/O devices. The MCU includes a 64-bit Control Store, two local stores, an arithmetic element, two busses to buffer memory, an unbuffered byte channel, and a priority interrupt system. Its cycle time is 150 nanoseconds (nsec).

The buffer store and SCU consists of eight fixed-priority Direct Memory Access (DMA), or Buffered, Channels communicating on a 150-nsec cycle basis with up to eight 32-bit-by-4K memories.

The SPAU is a highly parallel, high-speed arithmetic unit capable of performing complex signal processing operations such as FFT and recursive filtering. It is microprogrammed for flexibility in adapting signal processing algorithms. Four multiplies and six additions can be performed in 300 nsec.

SPE I/O and internal communications are provided by DMA buffer data channels, an unbuffered byte channel, and a priority interrupt system. The unbuffered byte channel called the Z Bus communicates both data and control information to all I/O devices. The unbuffered channel burst data rate is 2 million 16-bit words per second.

## PROBLEM STATUS

This is an interim report; work on these problems is continuing.

## AUTHORIZATION

NRL Problems B02-06 and B02-10
Project Nos. 3F21-241-601 (formerly WF-15-241-601),
2F00-241-601, XF-53-341-003, and XF-21-241-015-K152

# SIGNAL PROCESSING ELEMENT FUNCTIONAL DESCRIPTION (PRELIMINARY)

## PART 2 - Signal Processing Arithmetic Unit

### 1. INTRODUCTION

The NRL Signal Processing Element (SPE) is a high-performance signal processing facility for radar, sonar, and communication systems. The design of the SPE provides for efficient, flexible solutions to problems suited to digital signal processing machines. The SPE is intended to be compatible with the Navy All Applications Digital Computer (AADC) system now under development.

The SPE consists of the following elements:

Microprogrammed Control Unit (MCU)
Buffer memories
Storage Control Unit (SCU)
Input/output system
Signal Processing Arithmetic Unit (SPAU).

The SPE elements are implemented with "off-the-shelf" components. Bipolar monolithic storage devices and TTL Schottky family logic are used. Performance specifications include:

| | |
|---|---|
| MCU basic microinstruction | 150 nsec |
| Buffer memory cycle | 150 nsec |
| SPAU-equivalent complex operation (four multiplications and six adds) | 300 nsec |

Performance is compatible with projected AADC technology, and efficient operation can be expected under stand-alone or system-integrated conditions.

This report is a preliminary description of the SPAU.

## 1.1 Functional Description

The SPE is designed as a tool for processing digital data streams. The heart of the SPE is the MCU which serves as system supervisor and data organizer for the SPAU and other I/O devices in the system. Microprogrammed operations in the MCU process 16-bit-wide data accessed from 32-bit-wide buffer memories and control buffered and unbuffered I/O operations to and from SPE devices.

The SPAU performs special data processing operations such as FFT, recursive filtering, and correlation, under direction of the MCU. Parallel organization of fast multiply and add logic units allow for high-speed execution of these functions. Interfacing between the SPAU and MCU is via buffer memories and the I/O system. A partial list of SPAU operations (Macros) is contained in Appendix A.

It is the responsibility of the SCU to manage accesses to buffer memories by the elements of the SPE. The MCU, SPAU, and other buffered devices in the system access buffer memories independently under their own control, and the SCU resolves conflicts for buffer cycles on a priority basis.

The I/O system is designed to allow expansion of the SPE so that multiple MCU's and SPAU's can communicate and coordinate processing of increased data bandwidths.

Figure 1 is a block diagram of the SPE.

## 2. ORGANIZATION

### 2.1 Objectives

The SPAU has been designed to attain two primary objectives, high speed and efficiency, in the execution of signal processing algorithms. The former has been accomplished by using four parallel hardware multipliers and four adders in the section which performs arithmetic operations on the input data, and by concurrently generating memory addresses in a separate section which uses three parallel adders and three counters. High efficiency, that is the ability to keep most of the hardware busy most of the time, is accomplished by providing many data transfer options to the multipliers and adders.

During the design process, major emphasis has been placed on two signal processing algorithms: the fast Fourier Transform (FFT), and the second-order recursive filter. Another objective has been to provide flexibility for the efficient execution of other algorithms, such as data and spectrum weighting (Hanning) and vector and matrix operations. This overall flexibility has led to a wide control word (154 bits).

### 2.2 General Features

Figure 1 illustrates the relationship of the SPAU to the other elements in the SPE. The SPAU communicates with the Microprogrammed Control Units (MCU) by means of the Z bus and buffer memories. Input and output data areas residing in one or two buffer memories are assigned by an MCU each time the MCU issues a "macro" command to the SPAU. After receiving a macro, the SPAU operates in a stand-alone mode until it has finished the assigned task, then it sends an interrupt signal to the MCU which called it indicating that the macro has been completed.

In order to operate in this manner there are five functionally different sections combined within a SPAU, as illustrated in Figure 2.

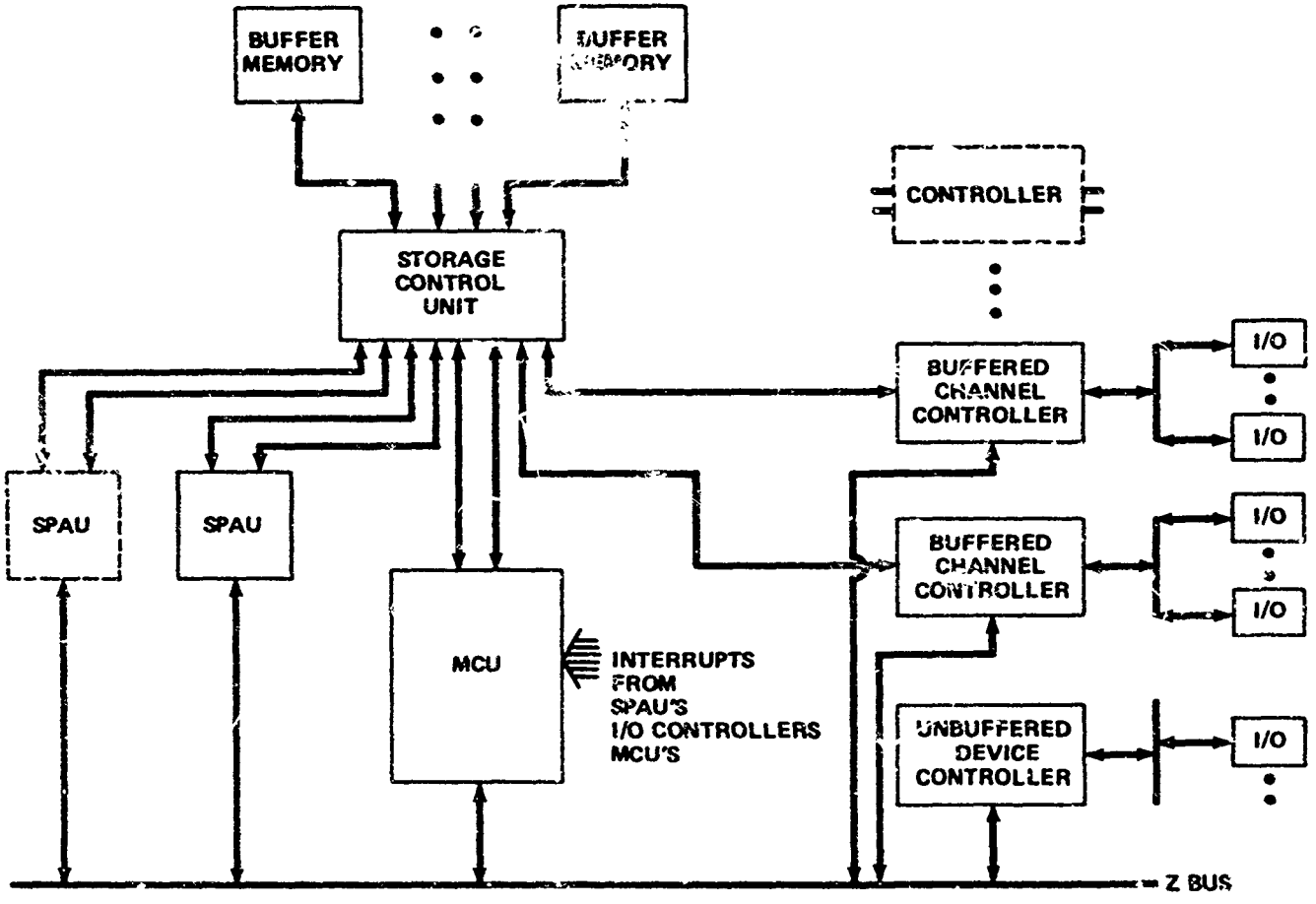
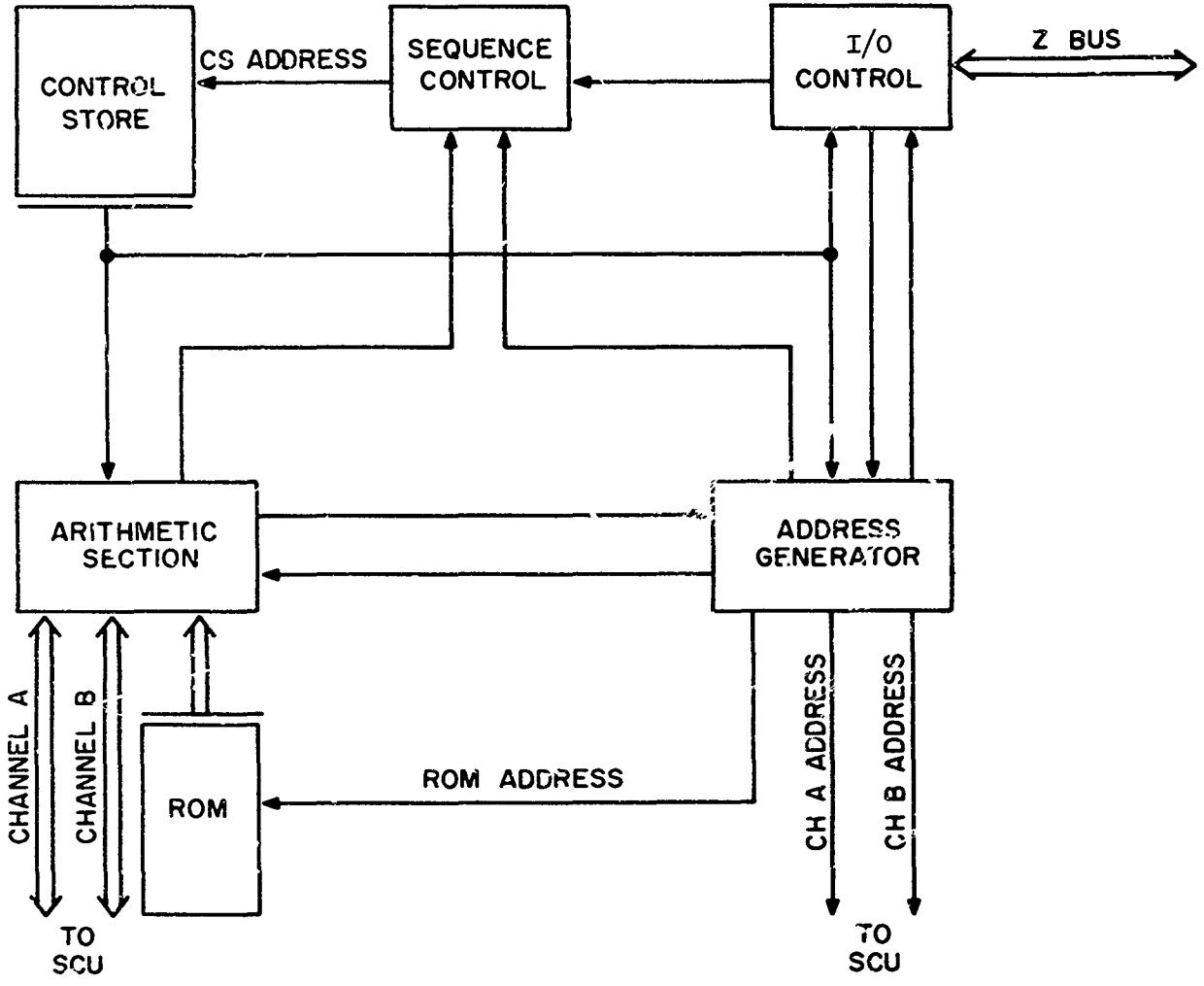



Fig. 1 - SPE system




SPAU—FUNCTIONAL UNIT BLOCK DIAGRAM

Figure 2

These are: the arithmetic section (AS), the address generator (AG), the sequence unit (SU), the control store (CS), and the input/output control Unit (IOCU). The AS contains four 16-bit multipliers; four 16-bit adders (arithmetic logic units); four each of input, product, and result registers; and four 16-word 16-bit local stores which are "ganged" in pairs (the two stores in a pair have common Read and Write addresses). The AG contains three adders, three counters, three output and three result registers, and a single 16-word 16-bit local store. Communication is provided between the AS and AG local stores in order to facilitate data dependent addressing. A read-only memory (ROM) contains 1025 sine and cosine coefficients, each 12 bits wide, for use in the FFT, plus often-used filter coefficients and other constants.

## 2.3 Operations

A SPAU operation may be initiated by an MCU sending an inquiry signal on the Z bus, and receiving a "not busy" reply from the SPAU. The MCU then sends a linkage message which includes the identity of the macro being requested, and its associated parameters. The message is transmitted via the IOCU to the W store in the AG and thence, as required, to the X and Y stores in the AS. The starting address of the particular macro in question is set up on the SU, and operation of both the AS and the AG begins.

The normal sequence of control is an unconditional step from one instruction to the next; however, this sequence can be altered by testing any one of fifteen other conditions in the AS and AG hardware, and transferring control to one of seven other successors. A new instruction is fetched every 150 nanoseconds (ns) unless a buffer memory access is denied to the SPAU, in which case the unit idles, re-requesting the memory access. Data are transferred to and from

buffer memories over two channels, denoted by A and B, of 32 bits width. The X and Y stores are each partitioned into 16-bit halves, $X_1$, $X_2$, and $Y_1$,$Y_2$, respectively, to operate with the 16-bit hardware of the AS.

Adder outputs may be loaded directly into result registers, R1 through R7, and multiplier outputs are always loaded into product registers, P1 through P4. In the case of P1 and P2, these may be the 16 more significant bits (MSB) or the 16 less significant bits (LSB) of the product, whereas in P3 and P4, they are always the MSB's. There are also four input registers, denoted by Z1 through Z4, which may be loaded from the ROM or from the source that is otherwise indicated (in the control word) for X and Y. The inputs to the AS multipliers and adders are obtained from X, Y, the Z registers, the P registers, and R1 through R4.

In the AG, the memory addresses are held in registers denoted by BARA, BARB, and RAR for channels A and B, and the ROM, respectively. Their contents are normally incremented by amounts contained in registers INCA, INCB, and INCR, respectively. The inputs to the AG adders are obtained from the address registers, the INC registers, the W store, or the literal field of the control word. The literal field and W are 16 bits wide; only the 11 least significant bits and the sign bit (the most significant bit) are used in the 12-bit AG hardware.

## 3. HARDWARE DESCRIPTION

The SPAU is composed of the following functional sections:

Control Store (CS)
Arithmetic Section (AS)
Address Generator (AG)
Sequence Unit (SU)
I/O Control Unit (IOCU)

### 3.1 Control Store

The SPAU control store is a read-only or read-mostly memory with a 150-nsec cycle time. The basic memory size is 256 words by 160 bits with expandability to 4096 words. The CS holds 160-bit microinstructions which control the operation of the SPAU. Input to the CS in the case of a writeable memory is via an independent buffered channel by which block loads of microprogram can be made from an external store. CS output is to a 160-bit command register (CR) which holds the microinstruction under execution. The address of the microinstruction to be fetched is provided by the Control Store Address Register.

### 3.2 Arithmetic Section (AS)

The arithmetic section of the SPAU contains the high speed arithmetic and storage elements necessary to provide the processing capability described in Section 2 of this report.

The AS contains four multipliers and four adders and their associated output registers. In addition, there are four coefficient registers and four local stores which can serve as input, output and intermediate data scratch pads. These arithmetic and storage elements are reconfigurable under program control so as to allow high hardware utilization and processing throughput for each SPAU application. The AS receives data from 32-bit wide buffer memories via two 32-bit wide channels. These two channels are bi-directional and thus serve as data output

channels as well. Consequently, data is always received and output in pairs of 16-bit words at a maximum rate of two 32-bit data transfers per instruction cycle.

Figure 3 is a block diagram of the SPAU arithmetic section.

### 3.2.1 AS Elements

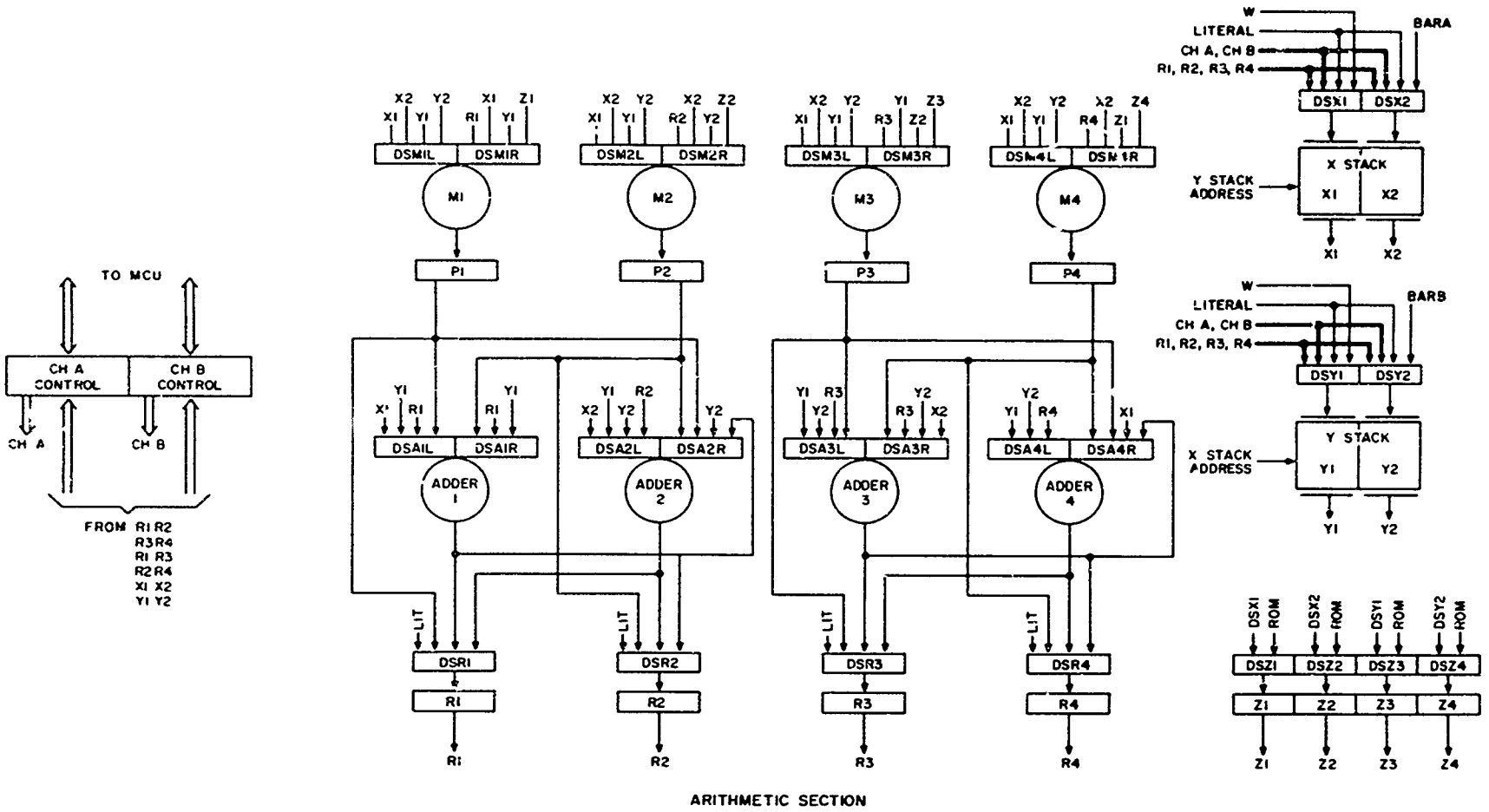
#### 3.2.1.1 Multipliers

There are four multipliers in the AS: M1, M2, M3 and M4. Each multiplier forms the product of two 16-bit numbers in two's complement arithmetic. The products are truncated to 16 bits in all the multipliers. M1 and M2 have the added capability of providing either the upper or lower 16-bit half of the product as specified in the microinstruction word. The time required to form a product is greater than the time available in one SPAU cycle, therefore the multipliers are designed to operate in a "pipelined" fashion with a throughput of one product every cycle and a time lag of two cycles. At the end of an instruction cycle then, one product will be output from inputs specified in the previous cycle and a partial product will be saved from inputs specified in the current cycle. A control bit in the microinstruction word allows the movement of products through the multiplier pipelines to be halted when necessary.

The multipliers have selectable inputs from the X and Y stores, from the Z coefficient registers and from the Result (R) registers. Each multiplier outputs to its own product register P.

#### 3.2.1.2 Arithmetic-Logic Units (ALU)

The AS contains four ALUs: A1, A2, A3 and A4. The ALUs perform two's complement arithmetic and logic operations on 16-bit data words. Each ALU can perform one ALU function (add, logical and, etc.) every instruction cycle. The functions available from each ALU are described in Section 4.0 of this report in the description of microinstruction control fields. The inputs to the ALUs are selectable and can come from the product register, the X and Y stores and the result registers. In addition, ALUs A2 and A4 have inputs directly from the output of A1 and




ARITHMETIC SECTION

Figure 3

A3, respectively. This allows two ALU operations to be cascaded in a single instruction cycle. The ALUs have outputs to the result registers and to the X and Y local stores.

3.2.1.3 Local Stores X and Y

The AS contains two 16-word by 32-bit local stores X and Y. X and Y can be independently addressed from fields in the control word. Each local store can be both read and written in each instruction cycle, and double addressing capability is provided whereby separate addresses can be specified for the read and write operations.

Each local store is divided vertically into two distinguishable stores each 16-bits wide. These are designated as X Upper (X1), X Lower (X2) and Y Upper (Y1), Y Lower (Y2). Although both halves of X or Y are not separately addressable, they are separately readable and writeable; that is, X1 can be accessed in a given instruction cycle while X2 is idle.

The local stores have inputs from Channels A and B, output registers R1, R2, R3 and R4, the literal field of the command register, and from the W store and buffer address registers in the Address Generator Section of the SPAU. When Channel A or B is a source to X or Y, then the left 16-bit half of the addressed buffer word inputs to X1 or Y1 and the right half to X2 or Y2. This same relationship holds in a buffer write operation.

Stores X and Y have outputs to Channels A and B, to the multiplier and ALU inputs, and to the W store in the Address Generator.

Data paths from X and Y to W provide for communication between the Arithmetic Section and Address Generator.

3.2.1.4 Coefficient Registers Z

Four 16-bit registers Z1, Z2, Z3 and Z4 are provided to receive preset coefficients from the ROM. They have outputs to the multipliers

only and are intended primarily to enable efficient processing of FFT and recursive filter algorithns. Z1, Z2, Z3 and Z4 have inputs from the first, second, third and fourth 12-bit quarters, respectively, of a 48-bit ROM word. The Z registers also have access to the same inputs as the X and Y stores via the X and Y data input selectors. As shown in Figure 3 and the control field descriptions, Z1 can receive the X1 input, Z2 the X2 input, Z3 the Y1 input and Z4 the Y2 input. This arrangement provides the ability to set the Z registers with computed or externally supplied data. The Z registers output to the right inputs of the multipliers.

3.2.1.5 Product Registers P

Each multiplier has an associated 16-bit register to receive its output product. These registers are P1, P2, P3 and P4. The P registers have outputs to the ALUs and to the Result registers.

3.2.1.6 Result Registers R

There are four 16-bit registers R1, R2, R3 and R4 provided to receive the outputs of ALU operations. Each R register can receive data from its associated ALU or from one adjacent ALU, from its associated product register or from the control word literal field.

The R registers have outputs back to the multiplier and ALU inputs to facilitate continuous sum or product type operations. They also output to the X and Y stores and to Channels A and B for direct data outputting. When outputting to buffer, the R registers output in pairs over the 32-bit channel width. The allowable pairs are R1R2, R3R4, R1R3 and R2R4.

3.3 Address Generator (AG)

The address generator section of the SPAU contains three independent address formation units which provide addresses for buffer Channels A and B and for the ROM. The AG operates concurrently with the AS, taking instructions from the same control words. A 16 word by 16-bit local

store in the AG provides scratch storage for data needed in computing address sequences. Each address formation unit is a simple structure composed of a 12-bit ALU and associated registers which hold addresses and address increments. In addition, there are three counter registers included in the AG for the purpose of allowing control of indexing and counting operations. Figure 4 is a block diagram of the Address Generator.

### 3.3.1 AG Elements

#### 3.3.1.1 ALUs

There are three ALUs in the AG which perform two's complement arithmetic and logic operations on 12-bit data words. The AG ALUs are designated A5, A6 and A7 to distinguish them from the four ALUs in the arithmetic section of the SPAU. Each ALU in the AG has three associated registers - a buffer address register and address increment register which have direct inputs to the ALU, and a result register which receives the ALU output. In addition, each ALU has inputs from local store W and from the command word literal field.

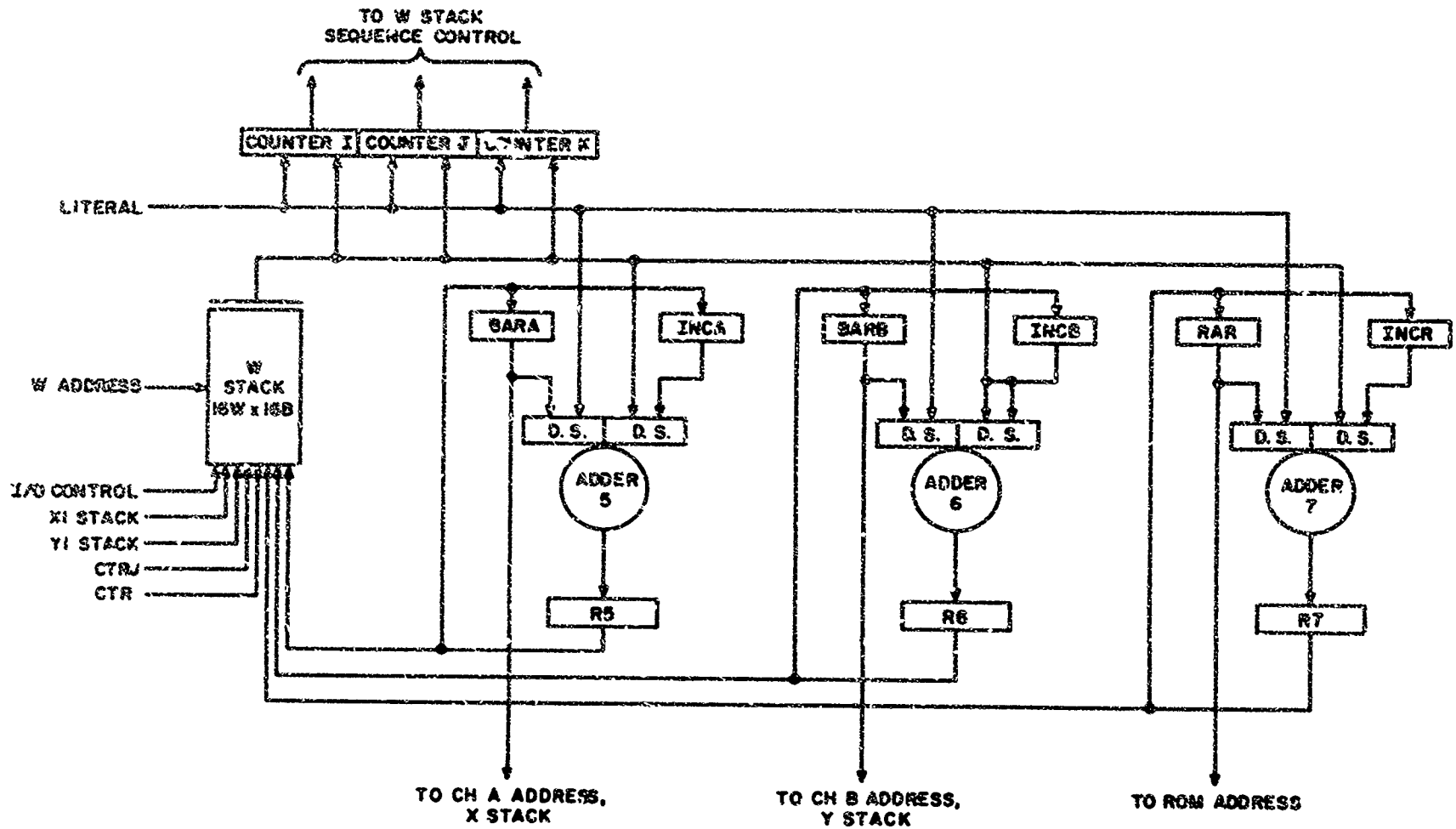
A5, A6 and A7 operations are described in Section 4 of this report.

#### 3.3.1.2 Result Registers R

Each ALU has its own associated 12-bit register to receive the output of ALU operations. These registers are designated R5, R6 and R7. Each R register has outputs to the W store and to its associated buffer address register and increment register.

#### 3.3.1.3 Buffer and ROM Address Registers BAR

Each ALU has its own associated address register. These supply addresses to buffers over Channel A and Channel B and to ROM, and they are designation BARA, BARB and RAR, respectively. The BARs are 12-bits wide and are intended primarily to hold the current buffer or ROM address being accessed by the arithmetic section. Normally, these registers are




ADDRESS GENERATOR

Figure 4

updated periodically by the ALUs using the values in the increment registers for incrementing or decrementing. Each BAR has an input from its associated ALU result register and outputs to the ALU.

### 3.3.1.4 Increment Registers INC

Each ALU has its own associated increment register - INCA, INCB and INCR. These are 12-bit, bi-directional shift registers intended to hold the address increment which is added to or subtracted from the BARs. Each INC register has an input from its associated ALU result register and outputs to the ALU. The INC registers are bi-directional shift registers to facilitate processing of certain algorithms such as FFT which require periodic altering of data address increments by factors of two.

### 3.3.1.5 Local Store W

Local store W is a 16 word by 16-bit store which provides scratch storage of data required in data address generation. The W store can be both read and written in a single instruction cycle but only at a single address. The W store has data paths to the AG ALUs and registers, and these paths are 12 bits wide. Twelve bits read from W comprise the low 11 data bits and the sign (most significant) bit. Twelve bits written into W will reside as the low 11 data bits with sign fill into bits 12 to 16 (the 5 MSBs). Inputs to W come from R5, R6 and R7, from two counters, CTRJ and CTRK, and from the X1 and Y1 stores in the SPAU arithmetic section.

There is also an input from the SPE Z bus via the SPAU I/O control unit. This enables communication of command and data messages between the SPAU and the MCU, and is described in more detail in Section 3.5 of this report.

The W store outputs to the AG ALUs, to counters CTRI, CTRJ and CTRK and X1 and Y1 in the AS.

### 3.3.1.6 Counters CTR

Three 12-bit counter registers are provided in the AG to facilitate control of indexing and counting operations. These counters can be set from the control word literal field or from the W store. A four-bit condition field in the SPAU microinstruction includes code points for testing and incrementing each counter register. A positive response to a specified counter test will change the control store sequence as specified by the Control Sequence field in the control word. The counters have data paths to the SPAU Sequence Unit for this purpose. Counters CTRJ and CTRK have outputs to W to enable saving of counter contents when necessary.

## 3.4 Sequence Unit (SU)

The Sequence Unit comprises that hardware which controls the sequence of SPAU operations by determining the order in which control words are fetched from the Control Store and by controlling the timing of arithmetic and data operations in other units of the SPAU. Sequence Unit operations are controlled by: Control Word bits 84 through 92 and 21 through 36; AS and AG element status; Buffer Memory cycle availability; and I/O control status.

### 3.4.1 Next Address Formation

The SU has a number of next address formations, both conditional and unconditional as well as an I/O controlled jump.

Possible control address sequences are:

a. Step - Increment Control Store Address Register (CSAR) by one.

b. Skip - Increment CSAR by two.

c. Save - Increment CSAR and save contents in Alternate CSAR (ACSAR).

d. Call - Jump to address specified in LIT field. CSAR plus one is saved in ACSAR.

e. Jump (LIT) - Jump to address specified in LIT field.

f. Jump (ACSAR) - Jump to address specified by the contents of the ACSAR.

g. Jump (R2) - Jump to address specified by the contents of R2.

h. Jump (R5) - Jump to address specified by the contents of R5.

i. I/O Controlled Jump - Jump to fixed control point address (0000). This jump is generated by an MCU command sent over the Z bus and initiates the sequence of instructions which sets up a specified macro operation.

All next address formations except I/O controlled jump can be conditioned upon the true or false states of certain SPAU conditions such as counter overflows, adder outputs zero or negative, etc. A description of the selectable conditions is given in Section 4 of this report.

The default next address is always a step.

### 3.4.2 Registers

The Sequence Unit uses two 12-bit address registers to enable efficient control of sequencing. The CSAR contains the current address which is sent to the control store. The CSAR is a counter register which can be incremented or loaded directly from the LIT field of the control register, from the ACSAR, from result registers R2 and R5, or cleared to zero by the I/O control unit.

The ACSAR acts as a temporary store of an address which can be directly transferred from or to the CSAR when required as in a CALL or JUMP (ACSAR) successor specification.

### 3.4.3 Sequence Timing

The Sequence Unit together with the system clock controls the occurrence of specified data operations in the SPAU. The 150 nsec basic clock cycle is subdivided into five subintervals called phases. Each phase is initiated by a clock pulse P/1, P/2, P/3, P/4 or P/5. Figure 5 is a chart showing the timing of events in the SPAU.

The Control Store (CS) cycle comprises those operations which address and fetch the control store and set its output into the command register. The control store is addressed and read one instruction ahead of the instruction in process. At P/1 the CSAR and ACSAR receive the addresses specified in the previous instruction, which are to be used in the following instruction. By P/4, the CS output is available and is set into the Command Register (CR). At P/5 CSAR is incremented by one in preparation for a step or skip successor specification for the next address. By the end of the cycle at P1, the CR successor and condition fields have been decoded, and the test condition states are available to control the sources to be closked into CSAR and ACSAR for the next cycle.

The Arithmetic Section cycle comprises data transfer and arithmetic operations specified for data processing operations. At P/1 of the AS cycle, those registers, local stores and arithmetic units which are specified as sources to other registers, local stores, arithmetic units or buffer channels are selected by the data selectors. Also, at P/1, specified source addresses for local stores X and Y are set into the X and Y address registers. By P/3, the X and Y outputs are available and are latched for the remainder of the cycle, and specified destination addresses are set into the X and Y address registers. By P/5, the outputs of sources, adders and multipliers, and buffer channels are available for setting into their

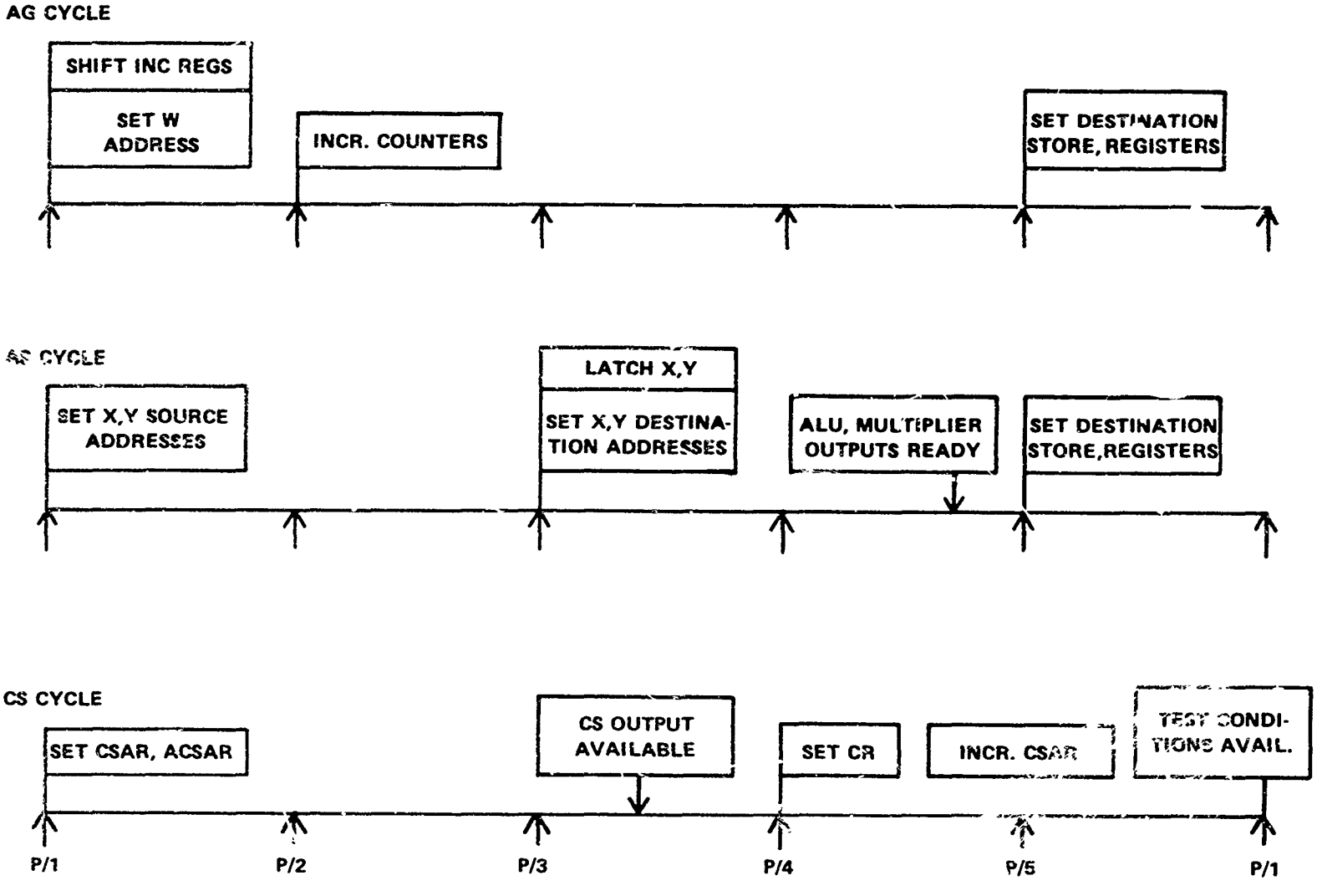



Fig. 5 EVENT TIMING

destination registers or local stores. P/5 initiates setting of data into their destinations which are available by P/1 for the start of the next cycle.

The Address generator cycle is similar to the AS cycle. Registers and local store W are selected and addressed at P/1 and set at P/5 with adder or source register outputs. Local store W being singly addressed does not have its address register reset at P/3. Also at P/1 of the AG cycle, INC register A, B or R is shifted left or right if specified. The counters I, J and K when tested are incremented at P/2.

## 3.5 I/O Control Unit (IOCU)

The I/O Control Unit comprises that logic which controls communication between the SPAU and MCU via the Z bus. The SPAU operates under control of an MCU, receiving commands to execute algorithms stored in the control memory and sending interrupts back upon completion of activity.

### 3.5.1 Interrupt Line

The interrupt line is the sole means by which the SPAU can directly establish communication with an MCU. The Z bus is under MCU control and the SPAU can put data on the Z bus only by a command from the MCU. The interrupt line outputs from a flip-flop in the I/O Control Unit which is set by a bit in the SPAU microinstruction word. The presence of this bit is an instruction causes the SPAU, after completion of that instruction to set the interrupt flip-flop in the IOCU and to suspend clock pulses to the AS, AG and SU. The SPAU remains stopped and the interrupt line raised pending receipt of I/O commands from the MCU over the Z bus. The interrupt flip-flop will be resent by the receipt of any MCU command.

### 3.5.2 Z Bus Communication

The IOCU continually monitors the 12 command lines of the Z bus. Eight of these lines carry the address of the device to which a command is being sent and another line (command line) indicates to all devices on the bus that a command is being sent. The IOCU is wired to recognize a command and a fixed device address as being its own. Two types of information are recognized by the I/O system. The first is control information and the second is data.

Control commands include requests for status, SPAU start and stop, etc. Control information goes directly to the IOCU. A 16-bit register in the IOCU contains all available SPAU status information. This register may be accessed by the MCU via the Z bus at any time whether the SPAU is running or stopped. All MCU outputs commands identified by the "Control" bit cause the IOCU to interpret the 16 data bits as a control command and to signal the SPAU to take appropriate action.

Data commands from the MCU can be both input and output commands and when the IOCU recognizes the "Data" bit it signals the AG to output or input a 16-bit W store word via the Z bus. Upon the transfer of each data word, if the "Continue" bit in the command is set to one, the W store address register is incremented to prepare for the next transfer. The W store address can be initialized to zero by a control command from the MCU.

Figure 6 is a functional diagram of the MCU-SPAU command interface.

### 3.5.3 MCU-SPAU Command Interface Operation

When the MCU has data ready in buffer memory for processing by the SPAU, the MCU can first send a control input command requesting the SPAU's status. The SPAU, upon recognition of the request, causes its

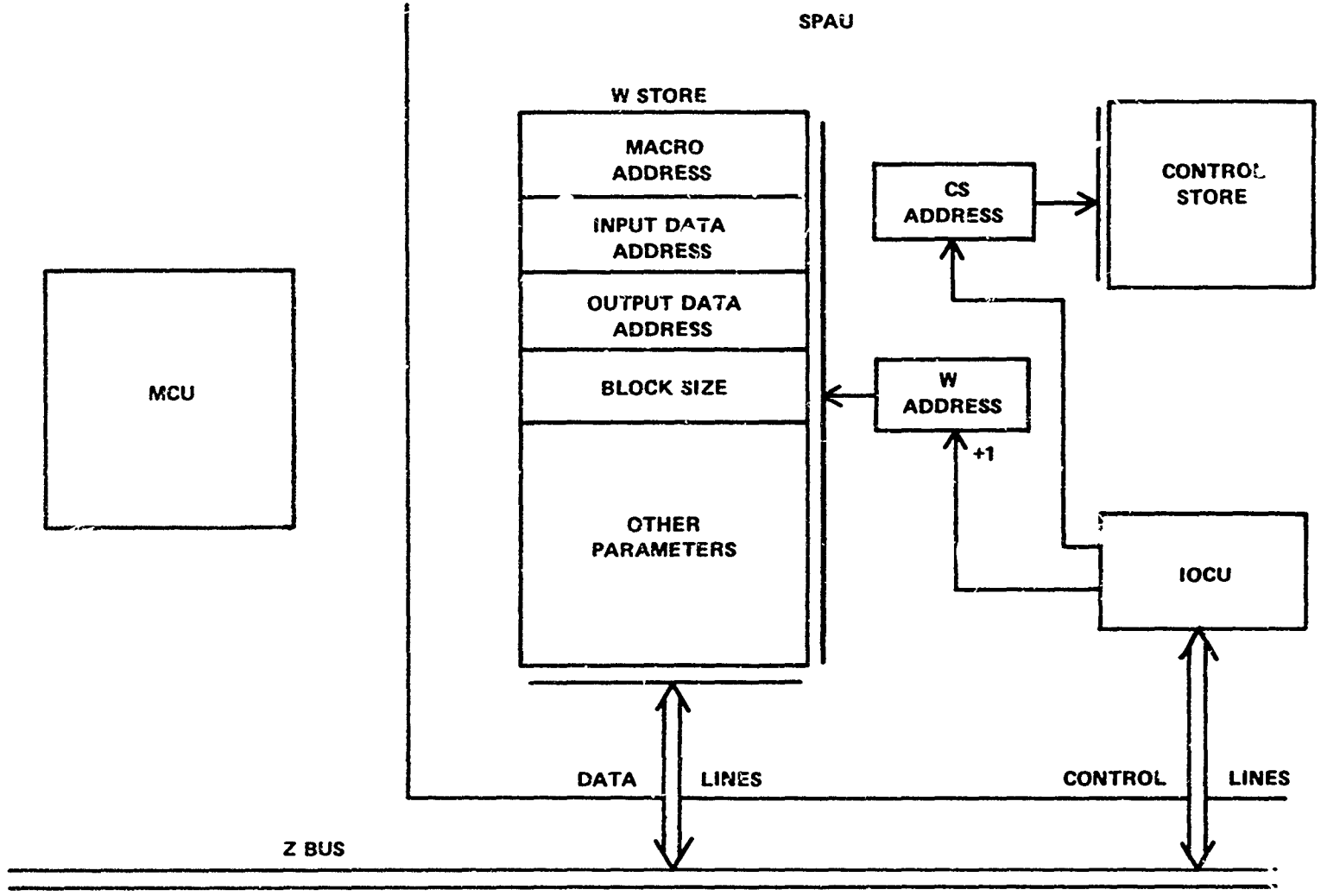



Fig. 6 MCU — SPAU COMMAND INTERFACE

status register to drive the Z bus data lines and drives the Acknowledge line to inform the MCU that the response is ready. Upon processing the received status, the MCU would then proceed with commands to set up the SPAU for the desired function. It is not necessary for the MCU to check the SPAU status prior to a command, but, if the SPAU is in a running mode, it will raise the Z bus Reject line in response to an output control command or to a data command.

To set up the SPAU for a processing function, the MCU should send at least one data word over the Z bus to the W store in the SPAU AG. This data word would contain an absolute or relative address code pointing to the SPAU control store location where the desired function resides. Additional data words would likely be transmitted defining parameters such as the starting addresses of data in buffer store, the number of data words, etc. Each output data command from the MCU which has a one on the Continue line will cause the W store address register to be incremented by one in preparation for the next data word.

At some point prior to finally issuing the start command to the SPAU, the MCU must send a command specifying which buffers are to be connected to channels A and B in the SPAU. This information goes into two 3-bit registers in the AG which, when concatenated with the AG address registers, form the full 15-bit addresses which go out to the SCU. The SPAU program has no effect on these registers and the buffer-to-channel connection remains fixed unless changed by the MCU.

When the MCU has transmitted the necessary parameters to the W store, a Start command can be issued to the SPAU. The Start command causes the SPAU CSAR to be set to zero and the Sequence Unit to resume SPAU operations. This trap to control store location 0000 provides for a system control point which can be programmed with instructions which retrieve the contents of W store location zero and process it to generate a computed jump, Jump (R5), to the first instruction of the requested routine.

The function routine proceeds to use the remaining data in W to set up the necessary registers and then carries out the indicated data processing.

Upon completion, a one in the Interrupt bit in the last instruction of the routine causes the SPAU to stop and the interrupt line to be energized. The interrupt line remains energized until the MCU responds with a command over the Z bus.

If desired, data can be stored in W by the routine for transmission back to the MCU. The procedure in this case is the same as described above except that the MCU would send data input rather than output commands, and W would be read rather than written.

Once the process is completed, another SPAU processing cycle may be initiated as above.

# 4. CONTROL WORD

## 4.1 Summary of Control Word Fields

| Control Register Bit Numbers | Field Function |
|---|---|
| 1- 4 | X Read Address |
| 5- 8 | X Write Address |
| 9-12 | Y Read Address |
| 13-16 | Y Write Address |
| 17-20 | W Address |
| 21-36 | Literal |
| 37-39 | W Source |
| 40-41 | Z Register Source |
| 42-44 | X1 Source |
| 45-47 | X2 Source |
| 48-50 | Y1 Source |
| 51-53 | Y2 Source |
| 54-56 | Literal Destination |
| 57 | Channel A and B Shift Control |
| 58-59 | INCA Register Shift Control |
| 60-61 | INCB Register Shift Control |
| 62-63 | INCR Register Shift Control |
| 64-66 | Counter Input |
| 67-69 | Channel A Source |
| 70-72 | Channel B Source |
| 73-74 | Adder 5 Destination |
| 75-76 | Adder 6 Destination |
| 77-78 | Adder 7 Destination |
| 79 | Increment Y Read |
| 80 | Increment Y Write |
| 81 | P1 and P2 Source |
| 82 | Multiplier Operations |
| 83 | I/O Interrupt |
| 84-87 | Condition |
| 88 | Condition Mode |

| Control Register Bit Numbers | Field Function |
|---|---|
| 89 | Jump on AOV |
| 90- 92 | Control Sequence |
| 93- 94 | Adder 1 Operation |
| 95- 96 | Adder 2 Operation |
| 97- 98 | Adder 3 Operation |
| 99-100 | Adder 4 Operation |
| 101-103 | Adder 5 Operation |
| 104-105 | Adder 6 Operation |
| 106-107 | Adder 7 Operation |
| 108-109 | R1 Source |
| 110-111 | R2 Source |
| 112-113 | R3 Source |
| 114-115 | R4 Source |
| 116-117 | A1 Left Source |
| 118-119 | A1 Right Source |
| 120-121 | A2 Left Source |
| 122-123 | A2 Right Source |
| 124-126 | A3 Left Source |
| 127-128 | A3 Right Source |
| 129-130 | A4 Left Source |
| 131-132 | A4 Right Source |
| 133 | A5 Left Source |
| 134 | A5 Right Source |
| 135 | A6 Left Source |
| 136 | A6 Right Source |
| 137 | A7 Left Source |
| 138 | A7 Right Source |
| 139-140 | M1 Left Source |
| 141-142 | M1 Right Source |
| 143-144 | M2 Left Source |
| 145-146 | M2 Right Source |
| 147-148 | M3 Left Source |
| 149-150 | M3 Right Source |
| 151-152 | M4 Left Source |
| 153-154 | M4 Right Source |

## 4.2 Control Field Definitions

4.2.1 The X Read Address identifies the location in the $X_1$ or $X_2$ store from which 16 bits may be read.

| Code Point | Store Address |
|---|---|
| 0 | 0 |
| . | . |
| . | . |
| . | . |
| 15 | 15 |

4.2.2 The X Write Address identifies the location in the $X_1$ or $X_2$ store into which 16 bits may be written.

| Code Point | Store Address |
|---|---|
| 0 | 0 |
| . | . |
| . | . |
| . | . |
| 15 | 15 |

4.2.3 The Y Read Address identifies the location in the $Y_1$ or $Y_2$ store from which 16 bits may be read.

| Code Point | Store Address |
|---|---|
| 0 | 0 |
| . | . |
| . | . |
| . | . |
| 15 | 15 |

4.2.4 The Y Write Address identifies the location in the $Y_1$ or $Y_2$ store into which 16 bits may be written.

| Code Point | Store Address |
|---|---|
| 0 | 0 |
| . | . |
| . | . |
| . | . |
| 15 | 15 |

4.2.5 The W Address identifies the location in the W store which may be used for reading and/or writing 12 bits.

| Code Point | Store Address |
|---|---|
| 0 | 0 |
| . | . |
| . | . |
| . | . |
| 15 | 15 |

4.2.6 The Literal field defines a 16-bit value which may be used as a source of data for one of the counters and/or for one of the X or Y stores, or one of the R registers, or for the ACSAR.

4.2.7 The W Source defines the source from which data shall be read into the W store.

| Code Point | Source |
|---|---|
| 0 | None |
| 1 | X1 |
| 2 | Y1 |
| 3 | CTRJ |
| 4 | CTRK |
| 5 | A5 |
| 6 | A6 |
| 7 | A7 |

4.2.8 The Z Register Source defines whether data will be read into the Z registers (Z1, Z2, Z3, Z4) from ROM, from the source defined for stores X and Y, or not at all.

| Code Point | Source |
|---|---|
| 0 | None |
| 1 | ROM |
| 2 | Load $Z_1$ and $Z_2$ from X source marked * below |
| 3 | Load $Z_3$ and $Z_4$ from Y source marked * below |

4.2.9 * The X1 Source defines the source from which 16 bits shall be read into the $X_1$ half of X.

| Code Point | Source |
|---|---|
| 0 | None |
| 1 | R1 |
| 2 | R2 |
| 3 | R3 |
| 4 | R4 |
| 5 | Upper half of Channel A (MSB) |
| 6 | Upper half of Channel B (MSB) |
| 7 | W |

4.2.10 * The X2 Source defines the source from which 16 bits shall be read into the $X_2$ half of X.

| Code Point | Source |
|---|---|
| 0 | None |
| 1 | R1 |
| 2 | R2 |
| 3 | R3 |
| 4 | R4 |
| 5 | Lower half of Channel A (LSB) |
| 6 | Lower half of Channel B (LSB) |
| 7 | BARA |

4.2.11 * The Y1 Source defines the source from which 16 bits shall be read into the Y1 half of Y.

| Code Point | Source |
|---|---|
| 0 | None |
| 1 | R1 |
| 2 | R2 |
| 3 | R3 |
| 4 | R4 |
| 5 | Upper half of Channel A (MSB) |
| 6 | Upper half of Channel B (MSB) |
| 7 | W |

4.2.12 * The Y2 Source defines the source from which 16 bits shall be read into theY2 half of Y.

| Code Point | Source |
|---|---|
| 0 | None |
| 1 | R1 |
| 2 | R2 |
| 3 | R3 |
| 4 | R4 |
| 5 | Lower half of Channel A (LSB) |
| 6 | Lower half of Channel B (LSB) |
| 7 | BARB |

4.2.13 * The Literal Destination defines the store or register into which the value of the literal field shall be copied.

| Code Point | Destination |
|---|---|
| 0 | None |
| 1 | X1 |
| 2 | X2 |
| 3 | Y1 |
| 4 | Y2 |
| 5 | Both R1 and R2 |
| 6 | Both R3 and R4 |
| 7 | ACSAR |

4.2.14 * The Channel A and B Shift Control defines whether or not the data read from Channels A and B will be shifted right one place during the Read operation.

| Code Point | Amount of Shift |
|---|---|
| 0 | None |
| 1 | Right one bit |

4.2.15 The INCA Register Shift Control defines the amount of shifting that shall be applied to data being read out of the INCA register.

| Code Point | Amount of Shift |
|---|---|
| 0 | None |
| 1 | Right one bit |
| 2 | Right two bits |
| 3 | Left one bit |

4.2.16 The INCB Register Shift Control defines the amount of shifting that shall be applied to data being read out of the INCB register.

| Code Point | Amount of Shift |
|---|---|
| 0 | None |
| 1 | Right one bit |
| 2 | Right two bits |
| 3 | Left one bit |

4.2.17 The INCR Register Shift Control defines the amount of shifting that shall be applied to data being read out of the INCR register.

| Code Point | Amount of Shift |
|---|---|
| 0 | None |
| 1 | Right one bit |
| 2 | Right two bits |
| 3 | Left one bit |

4.2.18 The Counter Input defines the source and destination of a counter loading operation.

| Code Point | Counter Operation |
|---|---|
| 0 | None |
| 1 | Addressed W to CTRI |
| 2 | Addressed W to CTRJ |
| 3 | Addressed W to CTRK |
| 4 | Literal to CTRI |
| 5 | Literal to CTRJ |
| 6 | Literal to CTRK |
| 7 | Unused |

4.2.19 The Channel A Source defines the source from which 32 bits shall be written into the Channel A buffer.

| Code Point | Source |
|---|---|
| 0 | None |
| 1 | R1 and R2 |
| 2 | R1 and R3 |
| 3 | R2 and R4 |
| 4 | R3 and R4 |
| 5 | X |
| 6 | Y |
| 7 | Unused |

4.2.20 The Channel B Source defines the source from which 32 bits shall be written into the Channel B buffer.

| Code Point | Source |
|---|---|
| 0 | None |
| 1 | R1 and R2 |
| 2 | R1 and R3 |
| 3 | R2 and R4 |
| 4 | R3 and R4 |
| 5 | X |
| 6 | Y |
| 7 | Unused |

4.2.21 The Adder 5 Destination defines the register(s) which shall receive the results of the Adder 5 operation.

| Code Point | Destination |
|---|---|
| 0 | None |
| 1 | R5 and BARA |
| 2 | R5 and INCA |
| 3 | R5 |

4.2.22 The Adder 6 Destination defines the register(s) which shall receive the results of the Adder 6 operation.

| Code Point | Destination |
|---|---|
| 0 | None |
| 1 | R6 and BARB |
| 2 | R6 and INCB |
| 3 | R6 |

4.2.23 The Adder 7 Destination defines the register(s) which shall receive the results of the Adder 7 operation.

| Code Point | Destination |
|---|---|
| 0 | None |
| 1 | R7 and RAR |
| 2 | R7 and INCR |
| 3 | R7 |

4.2.24 The Increment Y Read field indicates whether or not the Y store Read address shall be increased by one.

| Code Point | Definition |
|---|---|
| 0 | Bits 9-12 are the Y Read address |
| 1 | Increment previous Y Read address |

4.2.25 The Increment Y Write field indicates whether or not the Y store Write address shall be increased by one.

| Code Point | Definition |
|---|---|
| 0 | Bits 13-16 are the Y Write address |
| 1 | Increment previous Y Write address |

4.2.26 The P1 and P2 Source field indicates whether the P1 and P2 registers are loaded from the 16 least significant bits or the 16 most significant bits of multipliers M1 and M2, respectively.

| Code Point | Definition |
|---|---|
| 0 | Most significant bits |
| 1 | Least significant bits |

4.2.27 The Multiplier Operations field indicates whether all four multipliers (M1 through M4) shall propagate products to their P registers or whether they shall remain latched.

| Code Point | Operation |
|---|---|
| 0 | Propagate products |
| 1 | Hold products |

4.2.28 The I/O Interrupt field indicates whether the interrupt line from the SPAU to the MCU shall be in the raised state or the lowered state.

| Code Point | Definition |
|---|---|
| 0 | Lower the I/O interrupt |
| 1 | Raise the I/O interrupt |

4.2.29 The Condition specifies which one of the 16 tests shall be performed as a basis for determining the address (in control store) of the succeeding microinstruction.

| Code Point | Condition |
|---|---|
| 0 | Unconditional |
| 1 | CTRI overflow |
| 2 | CTRJ overflow |
| 3 | CTRK overflow |
| 4 | Adder 1 overflow |
| 5 | Adder 5 overflow |
| 6 | Adder 1 MSB |
| 7 | Adder 2 MSB |

| Code Point | Condition |
|---|---|
| 8 | Adder 5 MSB |
| 9 | Adder 7 MSB |
| 10 | Adder 1 LSB |
| 11 | Adder 2 LSB |
| 12 | Adder 5 LSB |
| 13 | Adder 7 LSB |
| 14 | Adder 1 all bits zero |
| 15 | Adder 2 all bits one |

4.2.30 The Condition Mode indicates whether the test specified by the Condition shall give a true or a false result as a basis for determining the address of the succeeding microinstruction.

| Code Point | Definition |
|---|---|
| 0 | True |
| 1 | False |

4.2.31 The Jump on AOV field indicates whether or not control shall jump to the address contained in ACSAR if an overflow occurs in Adder 1 or Adder 2 or Adder 3 or Adder 4.

| Code Point | Definition |
|---|---|
| 0 | Jump if AOV to ACSAR |
| 1 | Jump to Control Sequence successor |

4.2.32 The Control Sequence field indicates the source from which the address of the successor microinstruction shall be taken.

| Code Point | Source |
|---|---|
| 0 | Step (add one to the present address) |
| 1 | Skip (add two to the present address) |
| 2 | Save (add one to the present address and copy the present contents of CSAR into ACSAR) |
| 3 | Call (jump to ACSAR and copy the present contents of CSAR into ACSAR) |

| Code Point | Source |
|---|---|
| 4 | Jump to literal value as an address |
| 5 | Jump to ACSAR |
| 6 | Jump to R2 |
| 7 | Jump to R5 |

4.2.33 The Adder 1 Operation field defines the arithmetic or logical operation performed by Adder 1.

| Code Point | Operation |
|---|---|
| 0 | Left input + right input + 0 |
| 1 | Left input + right input + 1 |
| 2 | Left input - right input + 0 |
| 3 | Logical AND |

4.2.34 The Adder 2 Operation field defines the arithmetic or logical operation performed by Adder 2.

| Code Point | Operation |
|---|---|
| 0 | Left input - right input + 0 |
| 1 | Left input + right input + 0 |
| 2 | Left input + right input + 1 |
| 3 | Logical EQUIVALENCE |

4.2.35 The Adder 3 Operation field defines the arithmetic or logical operation performed by Adder 3.

| Code Point | Operation |
|---|---|
| 0 | Left input - right input + 0 |
| 1 | Left input + right input + 0 |
| 2 | Left input + right input + 1 |
| 3 | Logical EXCLUSIVE OR |

4.2.36 The Adder 4 Operation field defines the arithmetic or logical operation pe rmed by Adder 4.

| Code Point | Operation |
|---|---|
| 0 | Left input - right input + 0 |
| 1 | Left input + right input + 0 |
| 2 | Left input + right input + 1 |
| 3 | Logical EXCLUSIVE OR |

4.2.37 The Adder 5 Operation field defines the arithmetic or logical operation performed by Adder 5.

| Code Point | Operation |
|---|---|
| 0 | Left input + right input + 0 |
| 1 | Left input + right input + 1 |
| 2 | Left input - right input + 0 |
| 3 | Left input - right input + 1 |
| 4 | Left input + 0 |
| 5 | Logical INCLUSIVE OR |
| 6 | Logical AND |
| 7 | Unused |

4.2.38 The Adder 6 Operation field defines the arithmetic or logical operation performed by Adder 6.

| Code Point | Operation |
|---|---|
| 0 | Left input + right input + 0 |
| 1 | Left input + right input + 1 |
| 2 | Left input - right input + 0 |
| 3 | Left input - right input + 1 |

4.2.39 The Adder 7 Operation field defines the arithmetic or logical operation performed by Adder 7.

| Code Point | Operation |
|---|---|
| 0 | Left input + right input + 0 |
| 1 | Left input + right input + 1 |
| 2 | Left input - right input + 0 |
| 3 | Left input - right input + 1 |

4.2.40 The R1 Source defines the source from which 16 bits shall be read into R1.

| Code Point | Source |
|---|---|
| 0 | None or Literal |
| 1 | A1 |
| 2 | A2 |
| 3 | P1 |

4.2.41 The R2 Source defines the source from which 16 bits shall be read into R2.

| Code Point | Source |
|---|---|
| 0 | None or Literal |
| 1 | A1 |
| 2 | A2 |
| 3 | P2 |

4.2.42 The R3 Source defines the source from which 16 bits shall be read into R3.

| Code Point | Source |
|---|---|
| 0 | None or Literal |
| 1 | A3 |
| 2 | A4 |
| 3 | P3 |

4.2.43 The R4 Source defines the source from which 16 bits shall be read into R4.

| Code Point | Source |
|---|---|
| 0 | None or Literal |
| 1 | A3 |
| 2 | A4 |
| 3 | P4 |

4.2.44 The A1 Left Source defines the source from which 16 bits shall be read into the left input of Adder 1.

| Code Point | Source |
|---|---|
| 0 | P1 |
| 1 | X1 |
| 2 | Y1 |
| 3 | R1 |

4.2.45 The A1 Right Source defines the source from which 16 bits shall be read into the right input of Adder 1.

| Code Point | Source |
|---|---|
| 0 | P2 |
| 1 | Y1 |
| 2 | R1 |
| 3 | Unused |

4.2.46 The A2 Left Source defines the source from which 16 bits shall be read into the left input of Adder 2.

| Code Point | Source |
|---|---|
| 0 | Y1 |
| 1 | X2 |
| 2 | Y2 |
| 3 | R2 |

4.2.47 The A2 Right Source defines the source from which 16 bits shall be read into the right input of Adder 2.

| Code Point | Source |
|---|---|
| 0 | A1 |
| 1 | P1 |
| 2 | P2 |
| 3 | Y2 |

4.2.48 The A3 Left Source defines the source from which 16 bits shall be read into the left input of Adder 3.

| Code Point | Source |
|---|---|
| 0 | P3 |
| 1 | X2 |
| 2 | Y1 |
| 3 | Y2 |
| 4 | R3 |
| 5 | Unused |
| 6 | Unused |
| 7 | Unused |

4.2.49 The A3 Right Source defines the source from which 16 bits shall be read into the right input of Adder 3.

| Code Point | Source |
|---|---|
| 0 | P4 |
| 1 | R3 |
| 2 | Y2 |
| 3 | Unused |

4.2.50 The A4 Left Source defines the source from which 16 bits shall be read into the left input of Adder 4.

| Code Point | Source |
|---|---|
| 0 | Y2 |
| 1 | Y1 |
| 2 | R4 |
| 3 | Unused |

4.2.51 The A4 Right Source defines the source from which 16 bits shall be read into the right input of Adder 4.

| Code Point | Source |
|---|---|
| 0 | A3 |
| 1 | X1 |
| 2 | P3 |
| 3 | P4 |

4.2.52 The A5 Left Source defines the source from which 12 bits shall be read into the left side of Adder 5.

| Code Point | Source |
|---|---|
| 0 | BARA |
| 1 | Literal |

4.2.53 The A5 Right Source defines the source from which 12 bits shall be read into the right side of Adder 5.

| Code Point | Source |
|---|---|
| 0 | INCA |
| 1 | W |

4.2.54 The A6 Left Source defines the source from which 12 bits shall be read into the left side of Adder 6.

| Code Point | Source |
|---|---|
| 0 | BARB |
| 1 | Literal |

4.2.55 The A6 Right Source defines the source from which 12 bits shall be read into the right side of Adder 6.

| Code Point | Source |
|---|---|
| 0 | INCB |
| 1 | W |

4.2.56 The A7 Left Source defines the source from which 12 bits shall be read into the left input of Adder 7.

| Code Point | Source |
|---|---|
| 0 | RAR |
| 1 | Literal |

4.2.57 The A7 Right Source defines the source from which 12 bits shall be read into the right input of Adder 7.

| Code Point | Source |
|---|---|
| 0 | INCR |
| 1 | W |

4.2.58 The M1 Left Source defines the source from which 16 bits shall be read into the left input of Multiplier 1.

| Code Point | Source |
|---|---|
| 0 | X2 |
| 1 | X1 |
| 2 | Y1 |
| 3 | Y2 |

4.2.59 The M1 Right Source defines the source from which 16 bits shall be read into the right input of Multiplier 1.

| Code Point | Source |
|---|---|
| 0 | Z1 |
| 1 | X1 |
| 2 | Y1 |
| 3 | R1 |

4.2.60 The M2 Left Source defines the source from which 16 bits shall be read into the left input of Multiplier 2.

| Code Point | Source |
|---|---|
| 0 | X1 |
| 1 | X2 |
| 2 | Y2 |
| 3 | Y1 |

4.2.61 The M2 Right Source defines the source from which 16 bits shall be read into the right input of Multiplier 2.

| Code Point | Source |
|---|---|
| 0 | Z2 |
| 1 | Y2 |
| 2 | X2 |
| 3 | R2 |

4.2.62 The M3 Left Source defines the source from which 16 bits shall be read into the left input of Multiplier 3.

| Code Point | Source |
|---|---|
| 0 | X2 |
| 1 | X1 |
| 2 | Y1 |
| 3 | Y2 |

4.2.63 The M3 Right Source defines the source from which 16 bits shall be read into the right input of Multiplier 3.

| Code Point | Source |
|---|---|
| 0 | Z2 |
| 1 | Z3 |
| 2 | Y1 |
| 3 | R3 |

4.2.64 The M4 Left Source defines the source from which 16 bits shall be read into the left input of Multiplier 4.

| Code Point | Source |
|---|---|
| 0 | X1 |
| 1 | X2 |
| 2 | Y2 |
| 3 | Y1 |

4.2.65 The M4 Right Source defines the source from which 16 bits shall be read into the right input of Multiplier 4.

| Code Point | Source |
|---|---|
| 0 | Z1 |
| 1 | Z4 |
| 2 | Y2 |
| 3 | R4 |

APPENDIX A

SPAU MACROS

The macro is called from the MCU using a linkage which contains one or more of the following items:

1. Macro identification.
2. Starting address of data area (channel A, channel B, ROM).
3. Starting address of results area (channel A, channel B).
4. Size of the data area(s).
5. Replace/Add bit for results area.
6. Number of rows and columns of matrices.
7. Number of filter poles.
8. Number of multiplexed channels.
9. Data scale factor (block).
10. Other parameters associated with particular macros.

1. Foreward Fourier Transform - Considers consecutive input locations to be the real and imaginary components of a complex vector of dimension N. Denote the input vector by U: $u_r(0)$, $u_i(0)$, $u_r(1)$, $u_i(1)$, ..., $u_i(N-1)$. The result is a complex vector V, the transform of U: $v_r(0)$, $v_i(0)$, ... $v_r(N-1)$, $v_i(N-1)$.

2. Inverse Fourier Transform - Input is a complex vector, V. Output is its inverse Fourier transform, U.

3. Two Real Spectra - Considers the input complex vector V to be the transform of two real sequences contained in the Real and Imaginary parts of U. Output is half of each sequence's spectrum, packed due to conjugate symmetry, denoted by $A_1$ and $A_2$.

4. One Real Spectrum - Considers the input vector V to be the transform of a packed real input sequence contained in consecutive locations of U. Output is half of the spectrum of the real input sequence, denoted by C.

5. Conjugate Multiply - Considers two input sequences to be complex vectors, $V_1$ and $V_2$. Output is the product vector $J = V_1V_2^*$ (*denotes the complex conjugate).

6. Two Auto Spectra - Considers two input sequences to be $A_1$ and $A_2$, packed as in 3, above. Output is $|A|^2 = AA^*$, for both $A_1$ and $A_2$, in unpacked format: $|A_1(0)|^2$, $|A_2(0)|^2$, ..... $|A_1(N-1)|^2$, $|A_2(N-1)|^2$.

7. One Auto Spectrum - Considers the input sequence to be C, as output in 4, above. Output is $|C|^2 = CC^*$ in consecutive locations: $|C(0)|^2$, $|C(1)|^2$, ..... $|C(2N-1)|^2$.

8. Cross Spectrum - Considers the input sequence to be $A_1$ and $A_2$, packed as in 3, above. Output is the complex vector $Q = A_1A_2^*$: $q_r(0)$, $q_i(0)$, .....$q_r(N-1)$, $q_i(N-1)$.

9. Complex Multiply - Considers two input sequences to be complex vectors $V_1$ and $V_2$. Output is the product vector $P = V_1V_2$.

10. Real Multiply - Considers two input sequences to be real; $U_1$: $u_1(0)$, $u_1(1)$, .....$u_1(2N-1)$ and $U_2$: $u_2(0)$, $u_2(1)$, .....$u_2(2N-1)$. Output is the product of corresponding terms M: $m(0)$, $m(1)$, .....$m(2N-1)$, where $m(i) = u_1(i)\, u_2(i)$.

11. Scalar Product - Considers two input sequences to be real, $U_1$ and $U_2$, and forms the sum of products of corresponding terms. Output is $\sum m(i)$, i=0,...2N-1 as in 10, above.

12. Sum - Considers two input sequences to be real, $U_1$ and $U_2$, and forms the sums of corresponding terms. Output is S: $s(0)$, $s(1)$, ....$s(2N-1)$, where $s(i) = u_1(i) + u_2(i)$.

13. Accumulate - Forms the algebraic sum of all of the terms in the input sequence. (A variant may form the sum of every other term, for use with complex arrays).

14. Swap - Interchanges the contents of pairs of add and even locations in the input data array.

15. Weight Two Data - Consider the input sequence to be composed of two real sequences of length N in odd and even locations, respectively. Multiply both of them by $\frac{1}{2}(1-\cos 10\pi j/N)$ for $0 \le j \le .1N$, and by $\frac{1}{2}(1-\cos 10\pi (N-j)/N)$ for $.9N \le j \le N$.

16. Weight One Data - Consider the input to be a single real sequence of length 2N in consecutive locations, and multiply it by the coefficiencts described in 15, above.

17. Weight One Spectrum - Consider the input to be a complex vector V and combine adjacent terms, both Real and Imaginary (separately), as: $w_{r,i}(k) = -\frac{1}{4} v_{r,i}(k-1) + \frac{1}{2} v_{r,i}(k) - \frac{1}{4} v_{r,i}(k+1)$. Output is W: $w_r(0)$, $w_i(0)$, $w_r(1)$, .....$w_i(N-1)$.

18. Weight Two Spectra - Consider the input to be two packed spectra, $A_1$ and $A_2$, as in 3, above. For each spectrum combine the adjacent Real and Imaginary terms separately, as in 17, above, and output the results $B_1$ and $B_2$ in packed format.

19. Negate - Forms the two's complement of every item in the input data array.

20. Transpose - Consider the input sequence to be a real matrix of r rows and c columns, stored by columns. Output the transposed matrix by columns.

21. Invert - Form the inverse of an input square real matrix.

22. Matrix Multiply - Considers two input sequences to be real matrices of dimensions $r_1 \times c_1$ and $r_2 \times c_2$, with $c_1$ and $r_2$ given by a single value. Output is the matrix product, of dimension $r_1 \times c_2$.

23. Cascade N-Pole- Consider the input to be a multiplexed array of m real sequences. Filter each sequence with a cascade arrangement of N/2 2-pole filters. De-multiplex the output sequences.

24. Parallel N-Pole - Same as 23, above, except that a parallel arrangement is used instead of a cascade arrangement.

25. Rectangular to Polar - Considers the input sequence to be pairs of rectangular coordinates, x,y. Output is the corresponding pairs of polar coordinates.

26. Polar to Rectangular - Considers the input sequence to be pairs of polar coordinates, $r,\theta$. Output is the corresponding pairs of rectangular coordinates.

27. Search - The input is a single rectangular coordinate reference pair and an array of coordinate pairs as in 25, above. Output is the coordinates of the point(s) located at the minimum distance from the reference pair.

28. Key Search - Considers the input array to be a sequence of real values. Output consists of all input values for which a specified (composite) field is less than, equal to, or greater than, a given value. The number of such values is also output.

29. Polynomial evaluation.